Mobile Applicatios and Systems, Exercises "Local Datasources"

Prof. Dr.-Ing. Martin Schafföner

Summer Term 2022

1 Inspection of examples

Download the example source code archive from Moodle, unpack it and open it in IntelliJ by importing the top-level `build.gradle` file. Inspect the examples listed in the lecture slides and their appendix. Ask questions!

2 Data modelling

Create a class representing **Todos** in analogy to `de.thb.fbi.msr.maus.einkaufsliste.model.api.DataItem`. The Todo class should contain the following information:

1. a *unique identifier*,
2. the *title* or *name* of the todo,
3. a *description* of the todo,
4. whether the todo is *finished* or not,
5. whether the todo is *important / 'favorite'* or not,
6. the *due date*.

Use a `long` field for representing the due date in order to assist later persistence and transmission to a server.

3 List view with data access abstraction

1. Modify the created Todo list view from the previous exercise sheet so that it can display an arbitrary list of todos.

2. Design the access of the Todolist-Activity to the list of todos in such a way, that the technical details of the data access are hidden behind an interface. In your Activity, use an instance member variable of the interface type and assign it an instance of the concrete implementation class at Activity initialization time.

3. At least the name of the todo should be displayed in each list view item.

4. Try to also display the importance and finished status. You can use the images `res/drawable-mpi/star_grey.png` and `res/drawable-mpi/star_yellow.png` from the `Views` example project.

5. * The todos should be ordered according to the following criteria:
   a) unfinished todos before finished todos
   b) within the unfinished or finished todos the todos with closer due date should be displayed further up in the list than later todos.

6. Develop an implementation of the data access interface which initializes the todo list from static content, e.g. from a string array your application's resources. Write functionality is not required at this point.

4 Use of SQLite for data persistence

1. Implement your data access interface from exercise 3 which persists your todos in a local SQLite database with read and write access. Use the ROOM API.

2. Choose a suitable `ListAdapter` to display the database's content.

3. The `ListAdapter` should be initialized with a list of Todo objects.